Department of Computer Science and Engineering, University of Dhaka

Project Report

CSE-2211: Database Management Systems-I Lab

Project Title: Blood Bank Management System

Submitted By

Md. Emon Khan

Roll: 30

Submitted to

Dr. Md. Mustafizur Rahman, Professor

Dr. Muhammad Ibrahim, Assistant Professor

A. Blood Bank Management System

B. Project Area Major

1. Health
2. Hospital and Clinics
3. Blood Donation Foundation

C. Project Area Minor

1. University Blood Management
2. Healthcare Information System
3. Medical Inventory Management (This project can be extended to do this)

D. Brief Description About This Project

The Blood Donation and Healthcare Management System is a comprehensive database-driven solution designed to streamline and optimize the processes involved in blood donation, healthcare, and medical transactions. The system facilitates the efficient management of donor information, health histories, appointment scheduling, blood donations, inventory tracking, and patient requests. It serves as a centralized platform for healthcare professionals, donors, and patients to interact seamlessly and ensures the availability of blood resources for medical needs.

The system's core entities include the Person table, which captures individual details about blood donor/patient Information; the HealthHistory table, documenting health checkup records about donor/patient; and the Donation table, which records instances of blood donations. These tables are interconnected to maintain data integrity and provide a perfect view of an individual's health and blood donation activities.

Appointment scheduling is managed through the Appointment table by this user can appointment to give blood On the frontend side, while blood inventory is tracked in the **Inventory** table, ensuring real-time availability status. The **Request** table handles patient requests for blood, and the Receive table records the fulfillment of these requests. Users can make requests to get their desired blood group. **TransactionHistory** captures a log of donation and reception events, providing a historical perspective for analysis and auditing.

Automated triggers and functions enhance the system's functionality, **updating appointment and request statuses**, generating **transaction records**, and managing **inventory status** dynamically. The **UpdateExpiredInventoryStatus** procedure ensures the timely identification and handling of expired blood inventory.

This project aims to ensure transparency, accessibility, and efficiency in blood donation and healthcare operations. It empowers healthcare professionals to make informed decisions, enables donors to contribute effectively, and ensures timely and accurate responses to patient needs.

E. Detailed Description of the Database

The database will manage individuals, their health history, blood donation appointments, actual donations, blood inventory, donation requests, donation receptions, and transaction history. We can track and analyze blood-related transactions, supporting healthcare professionals in making informed decisions. Facilitates efficient management of blood inventory, and will ensure the availability of blood for patients in need. It will Enhance transparency and accountability through automated triggers and transaction logging. The system is designed to promote coordination between healthcare providers, blood donors, and patients, ensuring a collaborative and informed approach to healthcare management. Here is the details structure of this table

Table Details Description

1. Person Table:

- **Attributes:** person_id, full_name, age, gender, address, blood_group, phone_number, email, is_public_donor.
- **Description:** The Person table represents individuals within the system, storing personal details such as full name, age, gender, contact information, and blood group. The is_public_donor attribute indicates whether a person is a public blood donor.
- **Scopes:** This table serves as the foundation for tracking individuals involved in various aspects of the blood donation and healthcare system.

2. HealthHistory Table:

- **Attributes:** checkup_id, person_id, checkup_date, medical_condition.
- **Description:** The HealthHistory table records the health checkup history of individuals, associating each record with a person through the person_id foreign key. It includes details such as the checkup date and any diagnosed medical conditions.
- **Scopes:** Enables tracking and referencing the health status of individuals over time.

3. Appointment Table:

- **Attributes:** appointment_id, donor_id, appointment_date, status.
- **Description:** The Appointment table manages appointments for blood donations, linking each appointment to a person (donor) through the donor_id foreign key. The status field indicates whether the appointment is pending or the donation has occurred.
- **Scopes:** Facilitates the scheduling and tracking of blood donation appointments.

4. Donation Table:

- **Attributes:** donation_id, appointment_id, donor_id, donation_date, quantity, blood_group.
- **Description:** The Donation table records instances of blood donations, associating each donation with a specific appointment and donor. It includes details such as the donation date, quantity, and blood group.
- **Scopes:** Central to tracking blood donations, providing insights into donor activity.

5. Inventory Table:

- **Attributes:** inventory_id, donation_id, blood_group, quantity, collection_date, expiry_date, status, location.
- **Description:** The Inventory table manages the inventory of donated blood, linking each entry to a donation. It includes information such as blood group, quantity, collection date, expiry date, status (available, reserved, expired), and location.
- **Scopes:** Enables efficient blood inventory management and tracking.

6. Request Table:

- **Attributes:** request_id, patient_id, request_date, quantity, status, blood_group, hospital_name.

- **Description:** The Request table handles requests for blood from patients, associating each request with a specific patient. It includes details such as the request date, required quantity, status (pending or received), blood group, and the hospital making the request.
- **Scopes:** Facilitates the process of requesting and allocating blood to meet healthcare needs.

7. Receive Table:

- **Attributes:** receive_id, request_id, inventory_id, receive_date, quantity, total_cost.
- **Description:** The Receive table records instances of receiving blood from the inventory to fulfill specific requests. It is associated with both the request and the inventory, providing details such as the receive date, quantity received, and total cost.
- **Scopes:** Central to tracking the fulfillment of blood requests and associated costs.

8. TransactionHistory Table:

- **Attributes:** transaction_id, donation_id, receive_id, transaction_type, transaction_date, quantity.
- **Description:** The TransactionHistory table serves as a log for recording various transactions related to blood donations and receptions. It includes details such as transaction type (donation or receive), dates, and quantities involved.
- **Scopes:** Provides a historical record of blood-related transactions for auditing and analysis.

9. Functions and Triggers:

- **Functions:** The get_donation_id function retrieves the donation_id associated with a given inventory_id, facilitating data retrieval and consistency.
- **Triggers:** Multiple triggers automate updates in the system. The update_appointment_status trigger updates the status in the Appointment table after a donation, and the update_request_status trigger updates the status in the Request table after receiving blood. The donation_transaction_trigger and receive_transaction_trigger triggers generate transaction records based on donation and receive events. The update_inventory_status trigger adjusts inventory status based on the new quantity after a receive event.

10. UpdateExpiredInventoryStatus Procedure:

- **Description:** The UpdateExpiredInventoryStatus procedure updates the status of inventory items to "Expired" when the expiry_date is less than the current system date (SYSDATE). The COMMIT statement is used to make the changes permanent.
- **Scopes:** Ensures the timely management of expired blood inventory, maintaining the integrity of the database.

Table Relationship

Person Table and HealthHistory Table:

- Relationship: One-to-Many (1:N)
- Key Relationship Attribute: person_id (Person Table) and person_id (HealthHistory Table)
- Description: One person can have multiple health history records, but each health history record belongs to only one person.

Person Table and Appointment Table:

- Relationship: One-to-Many (1:N)
- Key Relationship Attribute: person_id (Person Table) and donor_id (Appointment Table)
- Description: One person can have multiple blood donation appointments, but each appointment is associated with only one person.

Person Table and Donation Table:

- Relationship: One-to-Many (1:N)
- Key Relationship Attribute: person_id (Person Table) and donor_id (Donation Table)
- Description: One person can make multiple blood donations, but each donation is associated with only one person.

Donation Table and Appointment Table:

- Relationship: One-to-One (N:1)
- Key Relationship Attribute: appointment_id (Donation Table) and appointment_id (Appointment Table)
- Description: One donation can be associated with one blood donation appointment

Donation Table and Inventory Table:

- Relationship: One-to-One (1:1)
- Key Relationship Attribute: donation_id (Donation Table) and donation_id (Inventory Table)

- Description: Each blood donation is directly linked to one inventory entry, ensuring a one-to-one correspondence.

Request Table and Person Table:

- Relationship: One-to-Many (1:N)
- Key Relationship Attribute: person_id (Person Table) and patient_id (Request Table)
- Description: One person can make multiple blood requests on behalf of different patients, but each request is associated with only one person.

Receive Table and Request Table:

- Relationship: One-to-One (1:1)
- Key Relationship Attribute: request_id (Receive Table) and request_id (Request Table)
- Description: Each blood reception event is directly linked to one blood request, ensuring a one-to-one correspondence.

Receive Table and Inventory Table:

- Relationship: Many-to-One (N:1)
- Key Relationship Attribute: inventory_id (Receive Table) and inventory_id (Inventory Table)
- Description: Multiple blood reception events can be associated with one inventory entry, but each entry corresponds to only one reception.

TransactionHistory Table and Donation Table:

- Relationship: Many-to-One (N:1)
- Key Relationship Attribute: donation_id (TransactionHistory Table) and donation_id (Donation Table)
- Description: Multiple transaction record is directly linked to one Donation event, ensuring a Many-to-one correspondence.

TransactionHistory Table and Receive Table:

- Relationship: One-to-One (1:1)
- Key Relationship Attribute: receive_id (TransactionHistory Table) and receive_id (Receive Table)
- Description: Each blood reception event is directly linked to one transaction record, ensuring a one-to-one correspondence.

F. Expected query outputs from project

1. Find the list of all the public Blood donors

This query will display all the public donors from the person table, who are always ready to donate blood.

Person Table

	PERSON_ID	FULL_NAME	AGE	GENDER	ADDRESS	BLOOD_GROUP	PHONE_NUMBER	EMAIL	IS_PUBLIC_DONOR
1	1	Emon Khan	21	Male	Mymensingh	O+	1234567890	emon.khan@email.com	Y
2	2	Rakin Afsar	16	Male	Mymensingh	A+	9876543210	rakin.afser@email.com	N
3	3	Tousif Khan	9	Male	Mymensingh	O+	5551112222	tousif.khan@email.com	Y
4	4	Zayan Hosain	5	Male	Mymensingh	AB−	9998887777	zayan.hosain@email.com	N
5	5	Nasif Khan	8	Male	Mymensingh	A+	4443332222	nasif.khan@email.com	Y

Query:

```
SELECT full_name, age, gender, blood_group
FROM Person
WHERE is_public_donor = 'Y';
```

Output

	FULL_NAME	AGE	GENDER	BLOOD_GROUP
1	Emon Khan	21	Male	O+
2	Tousif Khan	9	Male	O+
3	Nasif Khan	8	Male	A+

2. Find the Count of Total Number of O+ Public Blood Donors

This query will display all the public donors from the “Person” table, who are always ready to donate blood.

Query:

```
SELECT COUNT(*)
FROM Person
WHERE is_public_donor = 'Y' AND blood_group = 'O+';
```

Output

	COUNT(*)
1	2

3. Find the people who create an appointment and do not complete the donation process.

This query will find all the people who have not completed the Donation process yet

Appointment Table

	APPOINTMENT_ID	DONOR_ID	APPOINTMENT_DATE	STATUS
1	111	1	01-01-2023	Donated
2	222	2	01-01-2023	Donated
3	333	3	01-01-2023	Pending
4	444	4	01-01-2023	Pending
5	555	5	01-01-2023	Pending
6	545	1	01-02-2023	Donated

Query:

```
SELECT
  p.full_name,
  p.gender,
  p.email
FROM
  Person p
JOIN
  Appointment a ON p.person_id = a.donor_id
WHERE
  a.status = 'Pending';
```

Output

	FULL_NAME	GENDER	EMAIL
1	Tousif Khan	Male	tousif.khan@email.com
2	Zayan Hosain	Male	zayan.hosain@email.com
3	Nasif Khan	Male	nasif.khan@email.com

4. View the List of all the donors according to their donation count who have a person_id

This query will display all the donor information according to their donation count in descending order.

Query:

```
SELECT
  p.person_id,
  p.full_name,
  p.gender,
  p.email,
  COUNT(a.appointment_id) AS donation_count
FROM
  Person p
Left JOIN
  Appointment a ON p.person_id = a.donor_id AND a.status = 'Donated'
GROUP BY
  p.person_id, p.full_name, p.gender, p.email
ORDER BY
  donation_count DESC;
```

Output

	PERSON_ID	FULL_NAME	GENDER	EMAIL	DONATION_COUNT
1	1	Emon Khan	Male	emon.khan@email.com	2
2	2	Rakin Afsar	Male	rakin.afser@email.com	1
3	5	Nasif Khan	Male	nasif.khan@email.com	0
4	4	Zayan Hosain	Male	zayan.hosain@email.com	0
5	3	Tousif Khan	Male	tousif.khan@email.com	0
6	6	Iqra Islam	Female	iqra.islam@email.com	0

5. Find the person with the maximum amount of specific group blood donations

This query will display the information on who donated the maximum amount of blood

Query:

```
SELECT
  p.person_id,
  p.full_name,
  p.gender,
  p.email,
  SUM(d.quantity) AS total_o_positive_amount
```

```
FROM
  Person p
JOIN
  Donation d ON p.person_id = d.donor_id
WHERE
  p.blood_group = 'O+'
GROUP BY
  p.person_id, p.full_name, p.gender, p.email
ORDER BY
  total_o_positive_amount DESC
FETCH FIRST 1 ROW ONLY;
```

Output

	PERSON_ID	FULL_NAME	GENDER	EMAIL	TOTAL_O_POSITIVE_AMOUNT
1	1	Emon Khan	Male	emon.khan@email.com	7

6. Find the Total available Specific blood group in the Inventory

This Query will find all the available blood for a specific blood group name.

Query:

```
SELECT
  blood_group,
  SUM(quantity) AS total_available_amount
FROM
  Inventory
WHERE
  blood_group = 'O+'
  AND status = 'Available'
GROUP BY
  blood_group;
```

Inventory Table

	INVENTORY_ID	DONATION_ID	BLOOD_GROUP	QUANTITY	COLLECTION_DATE	EXPIRY_DATE	STATUS	LOCATION
1	1000	69	O+	1	02-01-2023	05-01-2023	Available	Hospital A
2	2000	79	A+	0	02-01-2023	04-01-2023	Reserved	Hospital B
3	3000	266	O+	4	02-02-2023	05-02-2023	Available	Hospital A

Output

	BLOOD_GROUP	TOTAL_AVAILABLE_AMOUNT
1	0+	5

7. View the List of Expired blood groups according to their donor name

Query:

```
SELECT
  i.inventory_id,
  i.blood_group,
  i.collection_date,
  p.full_name AS donor_name
FROM
  Inventory i
JOIN
  Donation d ON i.donation_id = d.donation_id
JOIN
  Person p ON d.donor_id = p.person_id
WHERE
  i.status = 'Expired';
```

Output

	INVENTORY_ID	BLOOD_GROUP	COLLECTION_DATE	DONOR_NAME
1	1000	0+	02-01-2023	Emon Khan
2	3000	0+	02-02-2023	Emon Khan

QuOutput

8. View all pending requests from the Request table according to the request dates

This query selects all columns from the Request table for pending requests, ordering the result set by the request date in ascending order

Query:

```
SELECT
  request_id,
  patient_id,
```

```
  request_date,
  quantity,
  status,
  blood_group,
  hospital_name
FROM
  Request
WHERE
  status = 'Pending'
ORDER BY
  request_date;
```

Request Table

	REQUEST_ID	PATIENT_ID	REQUEST_DATE	QUANTITY	STATUS	BLOOD_GROUP	HOSPITAL_NAME
1	534	3	02-01-2023	4	Received	O+	MMC
2	383	4	02-01-2023	2	Pending	AB-	CMH
3	414	5	02-01-2023	2	Received	A+	MMC
4	876	1	02-01-2023	4	Pending	O+	MMC
5	256	2	02-01-2023	2	Pending	A+	CMH
6	543	3	02-01-2023	2	Pending	O+	MMC

Output

	REQUEST_ID	PATIENT_ID	REQUEST_DATE	QUANTITY	STATUS	BLOOD_GROUP	HOSPITAL_NAME
1	383	4	02-01-2023	2	Pending	AB-	CMH
2	543	3	02-01-2023	2	Pending	O+	MMC
3	256	2	02-01-2023	2	Pending	A+	CMH
4	876	1	02-01-2023	4	Pending	O+	MMC

9. Retrieve the list of request IDs, names of individuals who have made the request, the requested amount, received amount, and total cost

Query:

```
SELECT
  r.request_id,
  p.full_name AS requester_name,
  r.request_date,
  r.quantity AS requested_amount,
  NVL(SUM(re.quantity), 0) AS received_amount,
  NVL(SUM(rc.total_cost), 0) AS total_cost
FROM
  Request r
JOIN
```

```
  Person p ON r.patient_id = p.person_id
LEFT JOIN
  Receive re ON r.request_id = re.request_id
LEFT JOIN
  Receive rc ON r.request_id = rc.request_id
GROUP BY
  r.request_id, p.full_name, r.request_date, r.quantity
ORDER BY
  r.request_id;
```

Output

	REQUEST_ID	REQUESTER_NAME	REQUEST_DATE	REQUESTED_AMOUNT	RECEIVED_AMOUNT	TOTAL_COST
1	256	Rakin Afsar	02-01-2023	2	0	0
2	383	Zayan Hosain	02-01-2023	2	0	0
3	414	Nasif Khan	02-01-2023	2	2	5000
4	534	Tousif Khan	02-01-2023	4	2	1000
5	543	Tousif Khan	02-01-2023	2	0	0
6	876	Emon Khan	02-01-2023	4	0	0

10. Display the total donation and receive amounts in one row for each day

In this query, the CASE statements inside the SUM function are used for conditional aggregation. They sum the quantity separately for 'Donation' and 'Receive' based on the transaction_type. The result will have one row for each day with the total donation and receive amounts.

Query:

```
SELECT
  transaction_date,
  SUM(CASE WHEN transaction_type = 'Donation' THEN quantity ELSE 0
END) AS total_donation,
  SUM(CASE WHEN transaction_type = 'Receive' THEN quantity ELSE 0
END) AS total_receive
FROM
  TransactionHistory
GROUP BY
  transaction_date
ORDER BY
  transaction_date;
```

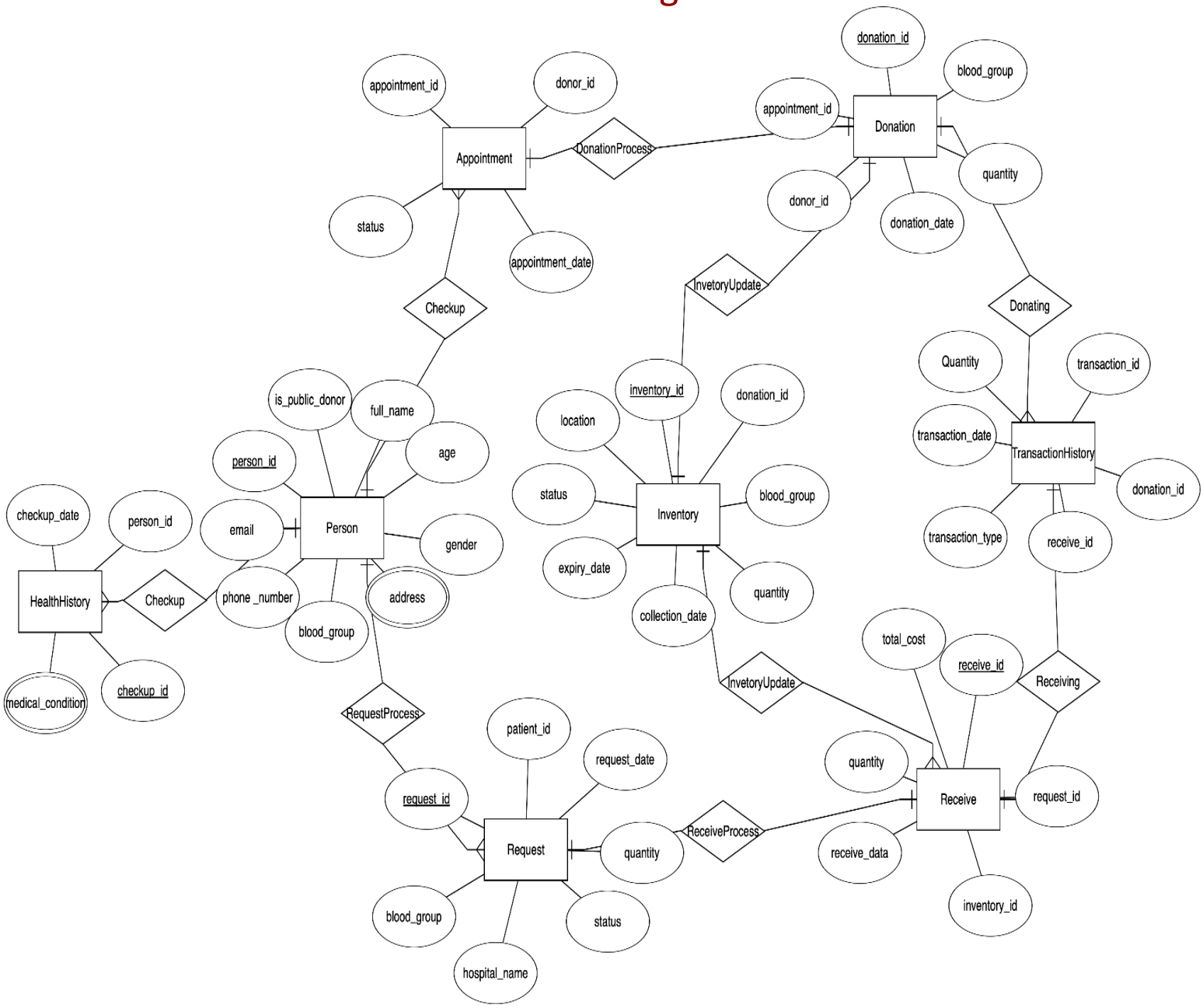
Output

	TRANSACTION_DATE	TOTAL_DONATION	TOTAL_RECEIVE
1	02-01-2023	5	4
2	02-02-2023	4	0

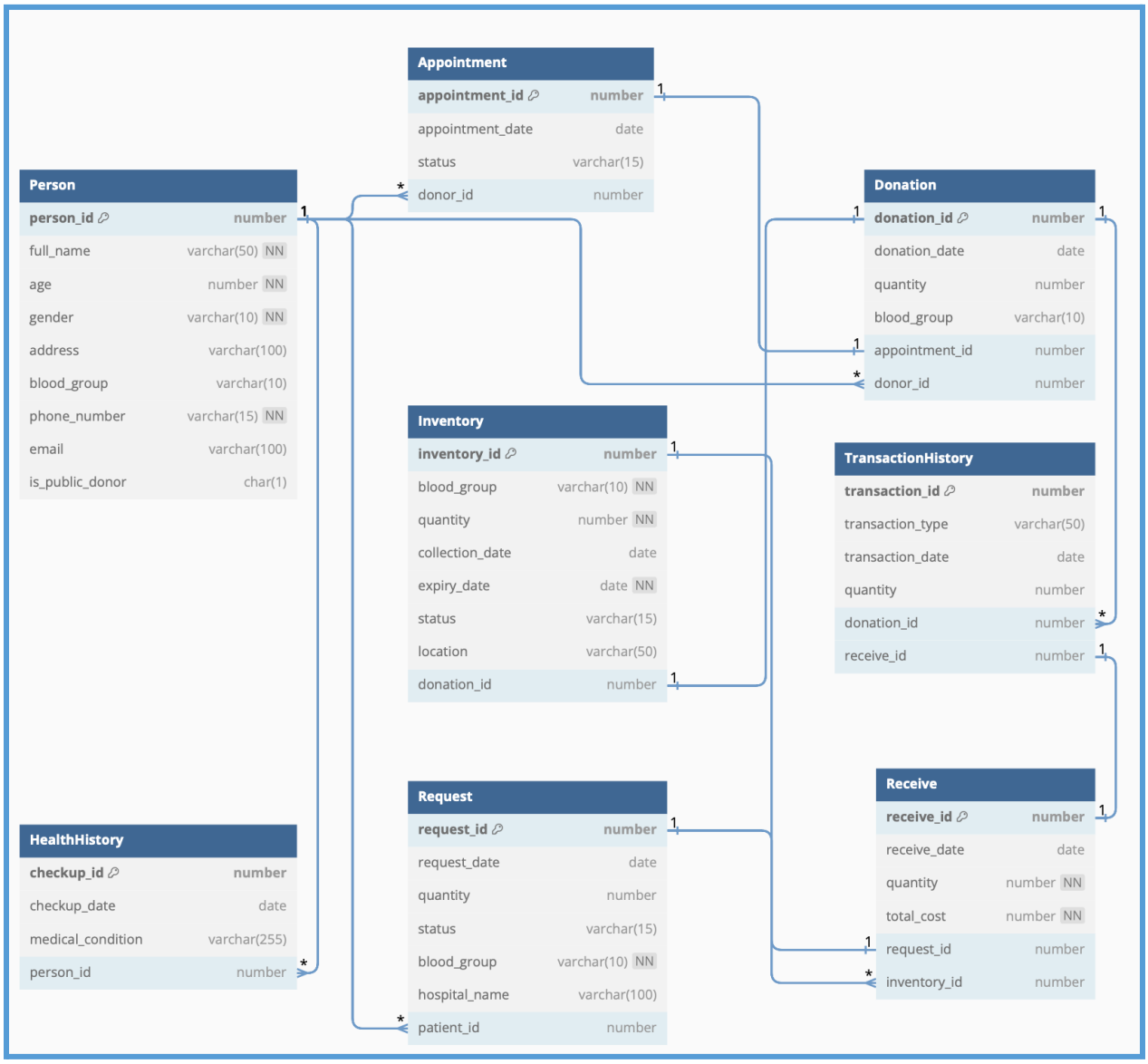
G. List of Tables with Schema

1. **Person** = (person_id, full_name, age, gender, address, blood_group, phone_number, email, is_public_donor);
2. **HealthHistory** = (checkup_id, person_id, checkup_date, medical_condition);
3. **Appointment** = (appointment_id, donor_id, appointment_date, status);
4. **Donation** = (donation_id, appointment_id, donor_id, donation_date, quantity, blood_group);
5. **Inventory** = (inventory_id, donation_id, blood_group, quantity, collection_date, expiry_date, status, location);
6. **Request** = (request_id, patient_id, request_date, quantity, status, blood_group, hospital_name);
7. **Receive** = (receive_id, request_id, inventory_id, receive_date, quantity, total_cost);
8. **TransactionHistory** = (transaction_id, donation_id, receive_id, transaction_type, transaction_date, quantity);

H-1. ER Diagram



H-2. Schema Diagram



I. List of Functional Dependencies to be maintained

Person:

```
person_id -> full_name, age, gender, address, blood_group,
phone_number, email, is_public_donor
```

HealthHistory:

```
checkup_id -> person_id, checkup_date, medical_condition
```

Appointment:

```
appointment_id -> donor_id, appointment_date, status
```

Donation:

```
donation_id -> appointment_id, donor_id, donation_date, quantity,
blood_group
```

Inventory:

```
inventory_id -> donation_id, blood_group, quantity, collection_date,
expiry_date, status, location
```

Request:

```
request_id -> patient_id, request_date, quantity, status,
blood_group, hospital_name
```

Receive:

```
receive_id -> request_id, inventory_id, receive_date, quantity,
total_cost
```

TransactionHistory:

```
transaction_id -> donation_id, receive_id, transaction_type,
transaction_date, quantity
```

J. Table schema

Person Table

S/N	Attribute	Data Type	Constraint	Comments
1	person_id	NUMBER	PK	Unique person ID can be NID/Birth Certificate ID
2	full_name	VARCHAR2(50)	NOT NULL	
3	age	NUMBER	NOT NULL	
4	gender	VARCHAR2(10)	NOT NULL	
5	address	VARCHAR2(100)		
6	blood_group	VARCHAR2(10)		Can be null. If a donor doesn't know currently
7	phone_number	VARCHAR2(15)	NOT NULL	
8	email	VARCHAR2(100)		
9	is_public_donor	CHAR(1)	CHECK (is_public_donor IN ('Y', 'N'))	If a person wants to become a public donor

Stores information about individuals. Includes personal details such as full name, age, gender, address, blood group, phone number, email, and whether the person is a public donor. Used to keep track of individuals and their personal information.

HealthHistory Table

S/N	Attribute	Data Type	Constraint	Comments
1	checkup_id	NUMBER	PK	
2	person_id	NUMBER	FK	
3	checkup_date	DATE		Automatic input

4	medical_condition	VARCHAR2(255)		

Stores health history records. Includes checkup ID, person ID, checkup date, and medical condition. Used to maintain a history of health checkups and medical conditions for individuals.

Appointment Table

S/N	Attribute	Data Type	Constraint	Comments
1	appointment_id	NUMBER	PK	
2	donor_id	NUMBER	FK	
3	appointment_date	DATE		Automatic input
4	status	VARCHAR2(15)		Pending/Donated

Stores information about donation appointments. Includes appointment ID, donor ID, appointment date, and status (Pending/Donated). Used to schedule and track donation appointments for individuals.

Donation Table

S/N	Attribute	Data Type	Constraint	Comments
1	donation_id	NUMBER	PK	
2	appointment_id	NUMBER	FK	
3	donor_id	NUMBER	PK	
4	donation_date	DATE		Automatic input
5	quantity	NUMBER	NOT NULL	
6	blood_group	VARCHAR2(10)	NOT NULL	

Records information about blood donations. Includes donation ID, appointment ID, donor ID, donation date, quantity, and blood group. Used to track blood donations and link them to specific appointments and donors.

Inventory Table

S/N	Attribute	Data Type	Constraint	Comments
1	inventory_id	NUMBER	PK	
2	donation_id	NUMBER	FK	
3	blood_group	VARCHAR2(10)	FK, NOT NULL	
4	quantity	NUMBER	NOT NULL	
5	collection_date	DATE		Automatic input
6	expiry_date	DATE	NOT NULL	
7	status	VARCHAR(15)		Available/Reserved/Expired
8	location	VARCHAR(50)		

Manages information about blood inventory. Includes inventory ID, donation ID, blood group, quantity, collection date, expiry date, status (Available/Reserved/Expired), and location. Used to monitor the availability, status, and location of blood units in the inventory.

Request Table

S/N	Attribute	Data Type	Constraint	Comments
1	request_id	NUMBER	PK	
2	patient_id	NUMBER	FK	
3	request_date	DATE		Automatic input

4	quantity	NUMBER		Constraint can be checked in frontend
5	status	VARCHAR2(15)		Pending/Received
6	blood_group	VARCHAR2(10)		Constraint can be checked in frontend
7	hospital_name	VARCHAR2(100)		

Records information about blood donation requests. Includes request ID, patient ID, request date, quantity, status (Pending/Received), blood group, and hospital name. Used to manage and track blood donation requests from patients.

Receive Table

S/N	Attribute	Data Type	Constraint	Comments
1	receive_id	NUMBER	PK	
2	request_id	NUMBER	FK	
3	inventory_id	NUMBER	FK	
4	receive_date	DATE		
5	quantity	NUMBER	NOT NULL	
6	total_cost	NUMBER	NOT NULL	

Stores information about received blood units. Includes receive ID, request ID, inventory ID, receive date, quantity, and total cost. Used to record and manage received blood units, linking them to specific requests and inventory items.

TransactionHistory Table

S/N	Attribute	Data Type	Constraint	Comments
1	transaction_id	NUMBER	PK	
2	donation_id	NUMBER	FK	
3	receive_id	NUMBER	FK	
4	transaction_type	VARCHAR2(50)		Donation/Receive
5	transaction_date	DATE		Automatic Input
6	quantity	NUMBER		

Records transaction history for donations and receives. Includes transaction ID, donation ID, receive ID, transaction type (Donation/Receive), transaction date, and quantity. Used to maintain a history of transactions related to blood donations and receives.

K. Table-level and project-level expected constraints

Table-Level Constraints:

Person

- Primary Key Constraint: person_id is the primary key.
- Check Constraint**:** is_public_donor can only have values 'Y' or 'N'.

HealthHistory

- Primary Key Constraint: checkup_id is the primary key.
- Foreign Key Constraint: person_id references Person(person_id).

Appointment:

- Primary Key Constraint: appointment_id is the primary key.
- Foreign Key Constraint: donor_id references Person(person_id).

Donation:

- Primary Key Constraint: donation_id is the primary key.
- Foreign Key Constraints:
  - appointment_id references Appointment(appointment_id).
  - donor_id references Person(person_id).

Inventory:

- Primary Key Constraint: inventory_id is the primary key.
- Foreign Key Constraint: donation_id references Donation(donation_id).
- Check Constraint: status can only have values 'Available', 'Reserved', or 'Expired'.

Request:

- Primary Key Constraint: request_id is the primary key.
- Foreign Key Constraint: patient_id references Person(person_id).
- Check Constraint**:** blood_group should not be NULL.

Receive:

- Primary Key Constraint: receive_id is the primary key.
- Foreign Key Constraints:
  - request_id references Request(request_id).
  - inventory_id references Inventory(inventory_id).

TransactionHistory:

- Primary Key Constraint: transaction_id is the primary key.
- Foreign Key Constraints:
  - donation_id references Donation(donation_id).
  - receive_id references Receive(receive_id).

Project-Level Constraints:

Consistency Across Blood Groups:

- Blood groups used in different tables (Person, Donation, Inventory, Request) are consistent and follow the same standard.

Referential Integrity:

- Foreign key relationships are maintained, meaning that values in referencing columns always point to valid, existing entries in the referenced columns.

Status Values Consistency:

- status values across different tables (Appointment, Inventory, Request) follow a consistent set of values.

Data Integrity:

- I use constraints to ensure data integrity, such as non-null values where required, valid date formats, and appropriate data types.

Consistency in Naming Conventions:

- Tried to maintain a consistent naming convention for tables and columns to enhance readability and understanding.

Transaction Type Validity:

- transaction_type in TransactionHistory is limited to valid types, such as 'Donation' or 'Receive'.

L. SQLs to implement the project with example outputs

Create Table:

```
CREATE TABLE Person (
  person_id NUMBER PRIMARY KEY,
  full_name VARCHAR2(50) NOT NULL,
  age NUMBER NOT NULL,
  gender VARCHAR2(10) NOT NULL,
  address VARCHAR2(100),
  blood_group VARCHAR2(10),
  phone_number VARCHAR2(15) NOT NULL,
  email VARCHAR2(100),
  is_public_donor CHAR(1) CHECK (is_public_donor IN ('Y', 'N'))
);

CREATE TABLE HealthHistory (
  checkup_id NUMBER PRIMARY KEY,
  person_id NUMBER,
  checkup_date DATE,
  medical_condition VARCHAR2(255),
  FOREIGN KEY (person_id) REFERENCES Person(person_id)
);

CREATE TABLE Appointment (
  appointment_id NUMBER PRIMARY KEY,
  donor_id NUMBER,
  appointment_date DATE,
  status VARCHAR2(15), -- Pending/Donated
  FOREIGN KEY (donor_id) REFERENCES Person(person_id)
```

```
);

CREATE TABLE Donation (
  donation_id NUMBER PRIMARY KEY,
  appointment_id NUMBER,
  donor_id NUMBER,
  donation_date DATE,
  quantity NUMBER NOT NULL,
  blood_group VARCHAR2(10) NOT NULL,
  FOREIGN KEY (appointment_id) REFERENCES Appointment(appointment_id),
  FOREIGN KEY (donor_id) REFERENCES Person(person_id)
);

CREATE TABLE Inventory (
  inventory_id NUMBER PRIMARY KEY,
  donation_id NUMBER,
  blood_group VARCHAR2(10) NOT NULL,
  quantity NUMBER NOT NULL,
  collection_date DATE,
  expiry_date DATE NOT NULL,
  status VARCHAR(15), -- Available/Reserved/Expired
  location VARCHAR(50),
  FOREIGN KEY (donation_id) REFERENCES Donation(donation_id)
);

CREATE TABLE Request (
  request_id NUMBER PRIMARY KEY,
  patient_id NUMBER,
  request_date DATE,
  quantity NUMBER,
  status VARCHAR2(15), -- Pending/Received
  blood_group VARCHAR2(10),
  hospital_name VARCHAR2(100),
  FOREIGN KEY (patient_id) REFERENCES Person(person_id)
);

CREATE TABLE Receive (
  receive_id NUMBER PRIMARY KEY,
  request_id NUMBER,
  inventory_id NUMBER,
  receive_date DATE,
  quantity NUMBER NOT NULL,
  total_cost NUMBER NOT NULL,
  FOREIGN KEY (request_id) REFERENCES Request(request_id),
  FOREIGN KEY (inventory_id) REFERENCES Inventory(inventory_id)
);
```

```
CREATE TABLE TransactionHistory (
  transaction_id NUMBER PRIMARY KEY,
  donation_id NUMBER,
  receive_id NUMBER,
  transaction_type VARCHAR2(50), --Donation/Receive
  transaction_date DATE,
  quantity NUMBER,
  FOREIGN KEY (donation_id) REFERENCES Donation(donation_id),
  FOREIGN KEY (receive_id) REFERENCES Receive(receive_id)
);
```

Function Used:

-- This function performs a simple SELECT query on the Inventory table using the provided inventory_id to retrieve the associated donation_id.
-- If a match is found, it returns the donation_id; otherwise, it returns NULL.

```
CREATE OR REPLACE FUNCTION get_donation_id(p_inventory_id NUMBER) RETURN
NUMBER IS
  v_donation_id NUMBER;
BEGIN
  SELECT donation_id
  INTO v_donation_id
  FROM Inventory
  WHERE inventory_id = p_inventory_id;

  RETURN v_donation_id;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
END get_donation_id;
/
```

-- Trigger to automatically updates the status in the Appointment table

```
CREATE OR REPLACE TRIGGER update_appointment_status
AFTER INSERT ON Donation
FOR EACH ROW
BEGIN
  UPDATE Appointment
  SET status = 'Donated'
  WHERE appointment_id = :new.appointment_id;
END;
/
```

```
-- Trigger to automatically updates the status in the Request table
```

```
CREATE OR REPLACE TRIGGER update_request_status
AFTER INSERT ON Receive
FOR EACH ROW
BEGIN
  UPDATE Request
  SET status = 'Received'
  WHERE request_id = :new.request_id;
END;
/
```

-- This trigger will generate a transaction record based on the donation_id and the current date/time
-- whenever a new donation record is inserted into the Donation table.

```
CREATE OR REPLACE TRIGGER donation_transaction_trigger
AFTER INSERT ON Donation
FOR EACH ROW
DECLARE
  v_transaction_id NUMBER;
BEGIN
  v_transaction_id := TO_NUMBER(TO_CHAR(SYSDATE, 'DDMMYYHH24MISS') ||
:new.donation_id);

  INSERT INTO TransactionHistory (
    transaction_id,
    donation_id,
    transaction_type,
    transaction_date,
    quantity
  ) VALUES (
    v_transaction_id,
    :new.donation_id,
    'Donation',
    :new.donation_date,
    :new.quantity
  );
END;
/
```

-- Trigger will insert data into the TransactionHistory table based on the Receive table:

```
CREATE OR REPLACE TRIGGER receive_transaction_trigger
AFTER INSERT ON Receive
FOR EACH ROW
DECLARE
  v_transaction_id NUMBER;
  v_donation_id NUMBER;
BEGIN
```

```
  v_donation_id := get_donation_id(:new.inventory_id); -- A Function Used
here

  v_transaction_id := TO_NUMBER(TO_CHAR(SYSDATE, 'DDMMYYHH24MISS') ||
:new.receive_id);

  INSERT INTO TransactionHistory (
    transaction_id,
    donation_id,
    receive_id,
    transaction_type,
    transaction_date,
    quantity
  ) VALUES (
    v_transaction_id,
    v_donation_id,
    :new.receive_id,
    'Receive',
    :new.receive_date,
    :new.quantity
  );
END;
/
```

Trigger Used

-- This trigger includes a check for the new quantity in the Inventory table after the update.
-- If the new quantity is 0, it updates the status to 'Reserved'.
-- If the new quantity is not 0, the status remains unchanged.

```
CREATE OR REPLACE TRIGGER update_inventory_status
AFTER INSERT ON Receive
FOR EACH ROW
DECLARE
  v_new_quantity NUMBER;
BEGIN
  UPDATE Inventory
  SET quantity = quantity - :NEW.quantity
  WHERE inventory_id = :NEW.inventory_id;

  SELECT quantity INTO v_new_quantity
  FROM Inventory
  WHERE inventory_id = :NEW.inventory_id;

  IF v_new_quantity = 0 THEN
    UPDATE Inventory
```

```
    SET status = 'Reserved'
    WHERE inventory_id = :NEW.inventory_id;
  END IF;
END;
/
```

Procedure Used

-- This procedure uses an UPDATE statement to change the status of inventory items to "Expired"
-- where the expiry_date is less than the current system date (SYSDATE).
-- The COMMIT statement is used to make the changes permanent.

```
CREATE OR REPLACE PROCEDURE UpdateExpiredInventoryStatus AS
BEGIN
  UPDATE Inventory
  SET status = 'Expired'
  WHERE expiry_date < SYSDATE and status = 'Available';

  COMMIT;
END UpdateExpiredInventoryStatus;
/
```

-- To Run Procedure UpdateExpiredInventoryStatus

```
BEGIN
  UpdateExpiredInventoryStatus;
END;
```

M. Finding the Normal Form (1st/2nd/3rd/3+ or BC-NF)

1. **Person Table:**
   All columns are atomic, and there are no repeating groups.
   It is already in the First Normal Form (1NF).
2. **HealthHistory Table:**
   All columns are atomic, and there are no repeating groups.
   It is already in the First Normal Form (1NF).
3. **Appointment Table:**
   All columns are atomic, and there are no repeating groups.
   It is already in the First Normal Form (1NF).
4. **Donation Table:**
   All columns are atomic, and there are no repeating groups.
   It is already in the First Normal Form (1NF).
5. **Inventory Table:**
   All columns are atomic, and there are no repeating groups.
   It is already in the First Normal Form (1NF).
6. **Request Table:**

All columns are atomic, and there are no repeating groups.
It is already in the First Normal Form (1NF).

7. **Receive Table:**
   All columns are atomic, and there are no repeating groups.
   It is already in the First Normal Form (1NF).
8. **TransactionHistory Table:**
   All columns are atomic, and there are no repeating groups.
   It is already in the First Normal Form (1NF).

N. Future Works and Conclusions

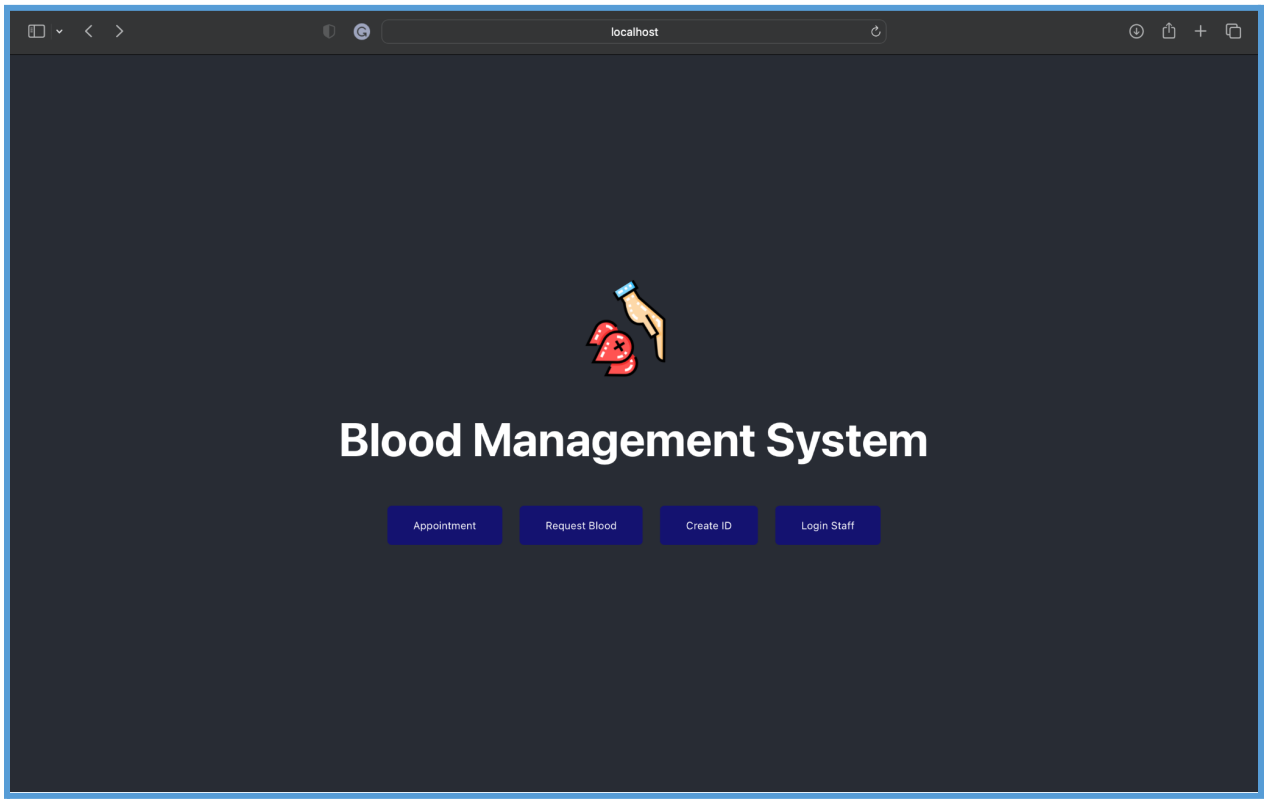
I have made a frontend **website** to take blood requests and donors can make appointments for donations. Other people can see the highest donor, and available blood group for individuals, can contribute to this community, also can create a user ID for further use. I used **reactJS** to make the frontend and **nodeJS** to make the backend server. The blood bank management system can be integrated with external systems, such as hospital management systems or regional/national blood donation databases, to streamline information exchange. It can be used to Implement advanced reporting and analytics features to generate insights from the data, helping blood banks make informed decisions about donation campaigns, inventory management, and resource allocation. I have a plan to develop a mobile application to allow donors to schedule appointments, receive notifications, and access their donation history. Additionally, a mobile app can facilitate real-time updates on available blood units for recipients. I tried to strengthen the security measures to ensure the confidentiality and integrity of sensitive health data.

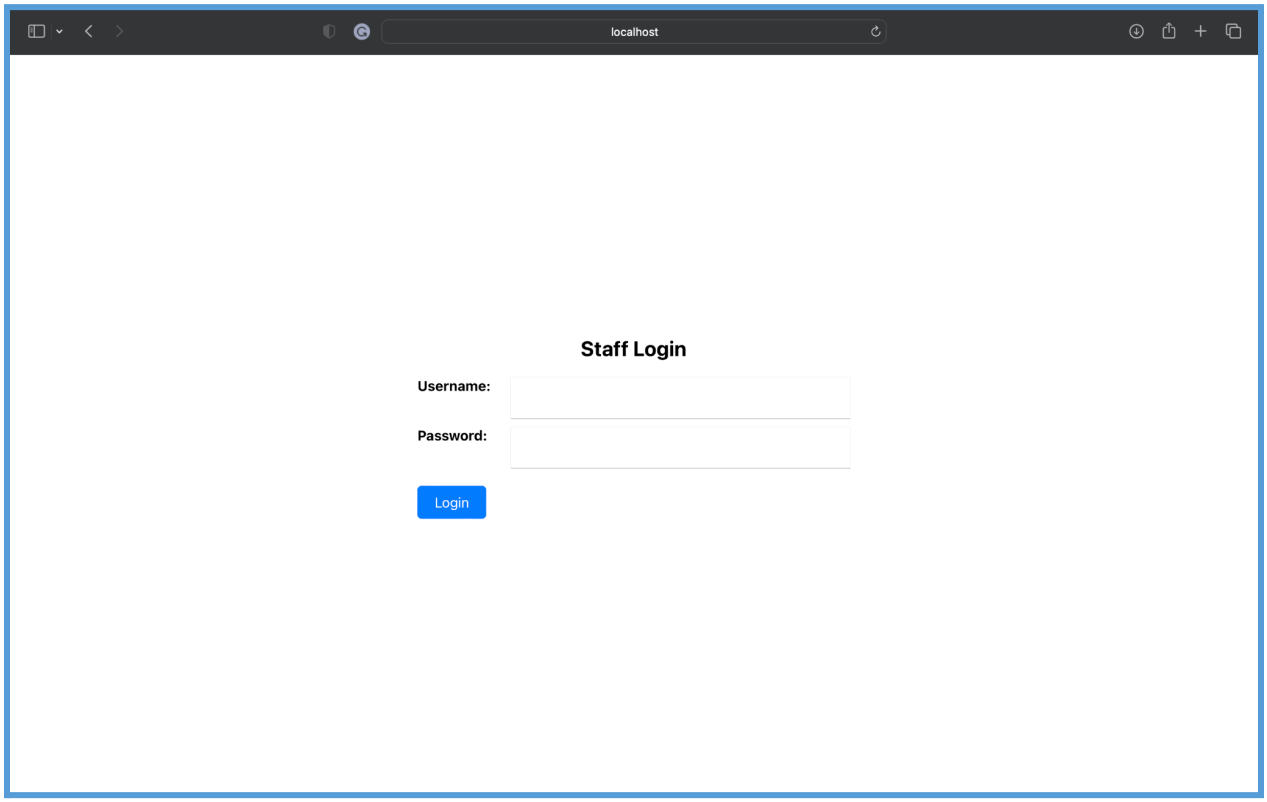
The blood bank management system facilitates a more efficient and organized blood donation process. Donors can easily schedule appointments, and the system helps in managing the entire donation lifecycle. By streamlining blood inventory management and donation request processes, the system contributes to better patient care by ensuring the availability of required blood units when needed. It can be expanded to include additional features or integrated with other healthcare systems. By ensuring a steady and well-managed blood supply, the system contributes to public health initiatives, emergency response preparedness, and overall healthcare infrastructure.

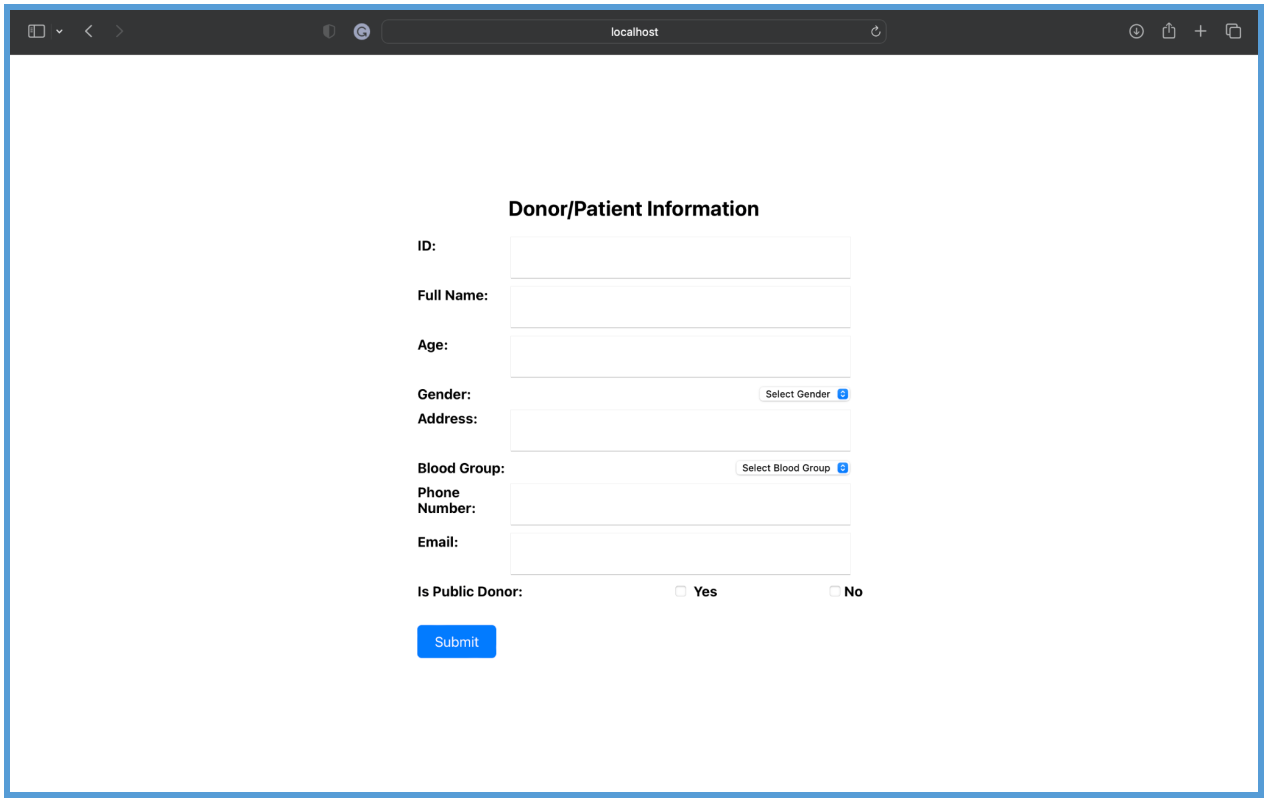
Resource

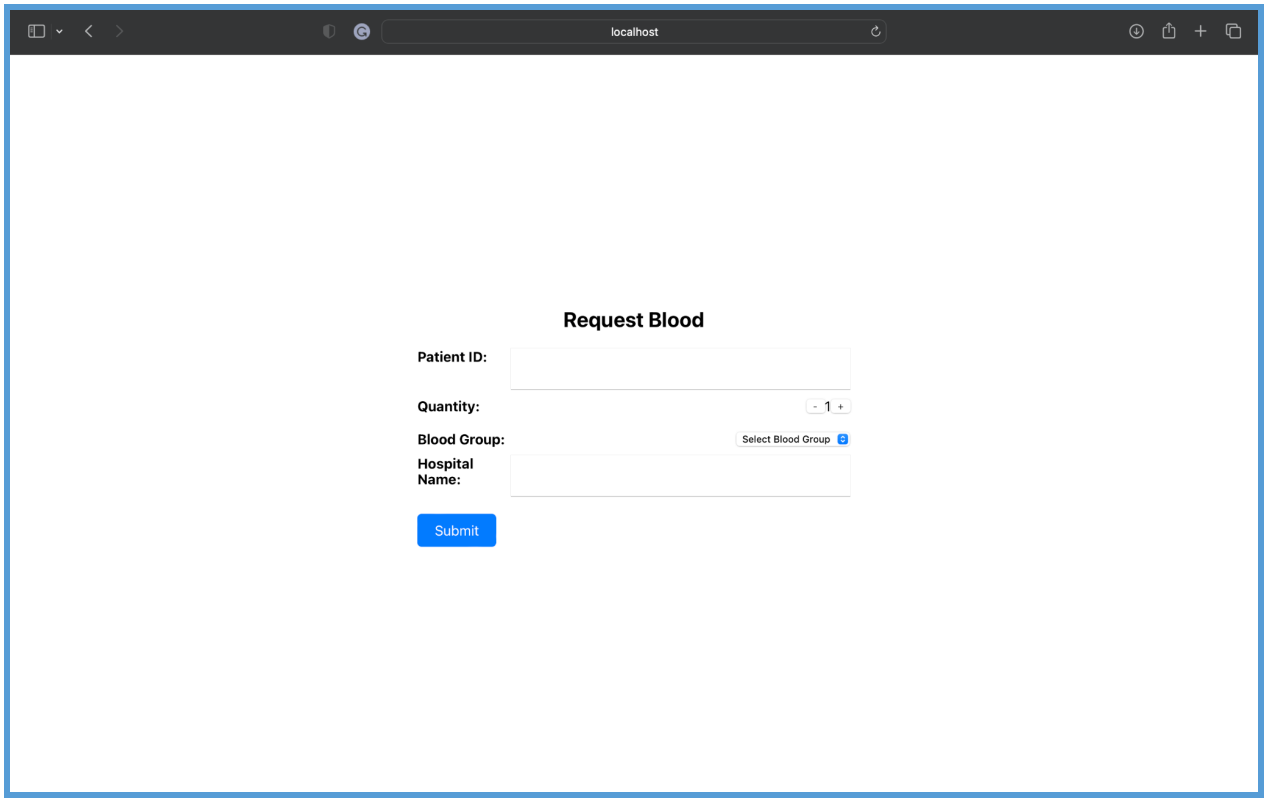
1. Project Github:https://github.com/ignite312/Blood-Bank-Management-System
2. Demo Website frontend: https://ignite312.github.io/Blood-Bank-Management-System/
3. Frontend : **reactJS**, Backend : **nodeJS**, Database: **Oracle**
4. The tool used to create Schema Diagram: https://dbdiagram.io/d
5. The tool used to create ER Diagram: https://erdplus.com/standalone
6. Frontend logo: https://www.flaticon.com/

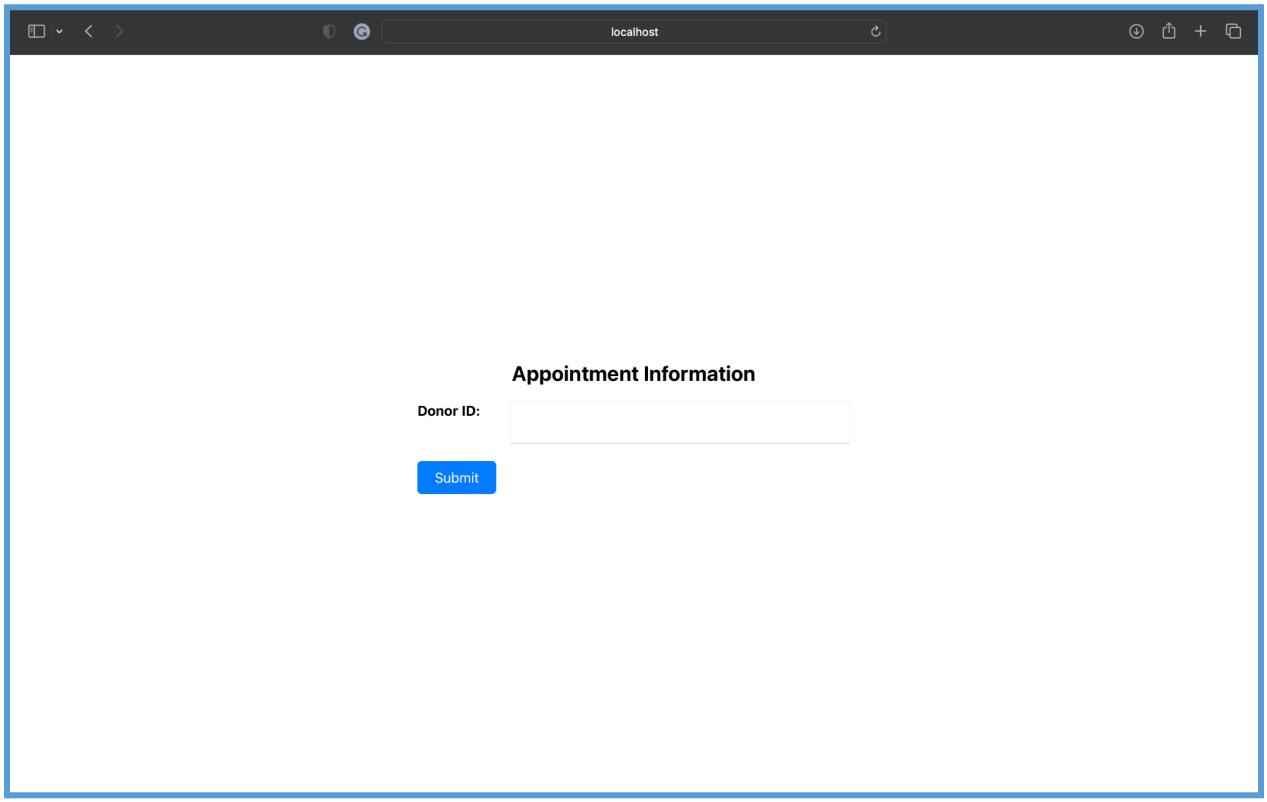
Frontend UI Development






localhost
Donor/Patient Information
ID:
Full Name:
Age:
Gender:
Select Gender
Address:
Blood Group:
Select Blood Group
Phone Number:
Email:
Is Public Donor:
Yes
No
Submit


localhost
Request Blood
Patient ID:
Quantity:
- 1 +
Blood Group:
Select Blood Group
Hospital Name:
Submit


localhost
Appointment Information
Donor ID:
Submit